Department of Information Systems

Subject : Requirement Engineering

Functional Requirement (FR)

Non Functional Requirements (NFR)

IS184309, 3 sks

Dr. Rarasmaya Indraswari, S.Kom.

Learning Objectives

- To introduce types of requirement
- To comprehend the differences between functional and non functional requirements (FRs & NFRs)

Outline

- ✓ FR
- ✓ NFR
- ✓ Example

What is Requirements?

Requirements are distinguished in two parts:

- Functional requirements, which describe what the system does
- Non-functional requirements, which describe how the system performs the functional requirements (in terms of performance, security, usability etc).

Functional Requirements must be in accordance with the ones described in the Tender and Contract Documents, unless mutually agreed and documented. Use the relevant functional requirements presented in the Tender or Contract Document as a guide during the Requirements gathering phase

The non-functional requirements for a system are typically constraints on the functional requirements – that is, not what the system does, but how it does it (how quickly, how efficiently, how easily from the user's perspective, etc.). In other words Non-Functional Requirements describe qualitative perspectives of the system. Other non-functional requirements may be required characteristics that are not part of the system's functionality, e.g., conformance with legal requirements, scalability, interoperability, etc.

Project management issues (costs, time, schedule) are often considered as non-functional requirements as well

Examples of Requirements

The product will support multiple human languages

The persistence will be handled by a relational database

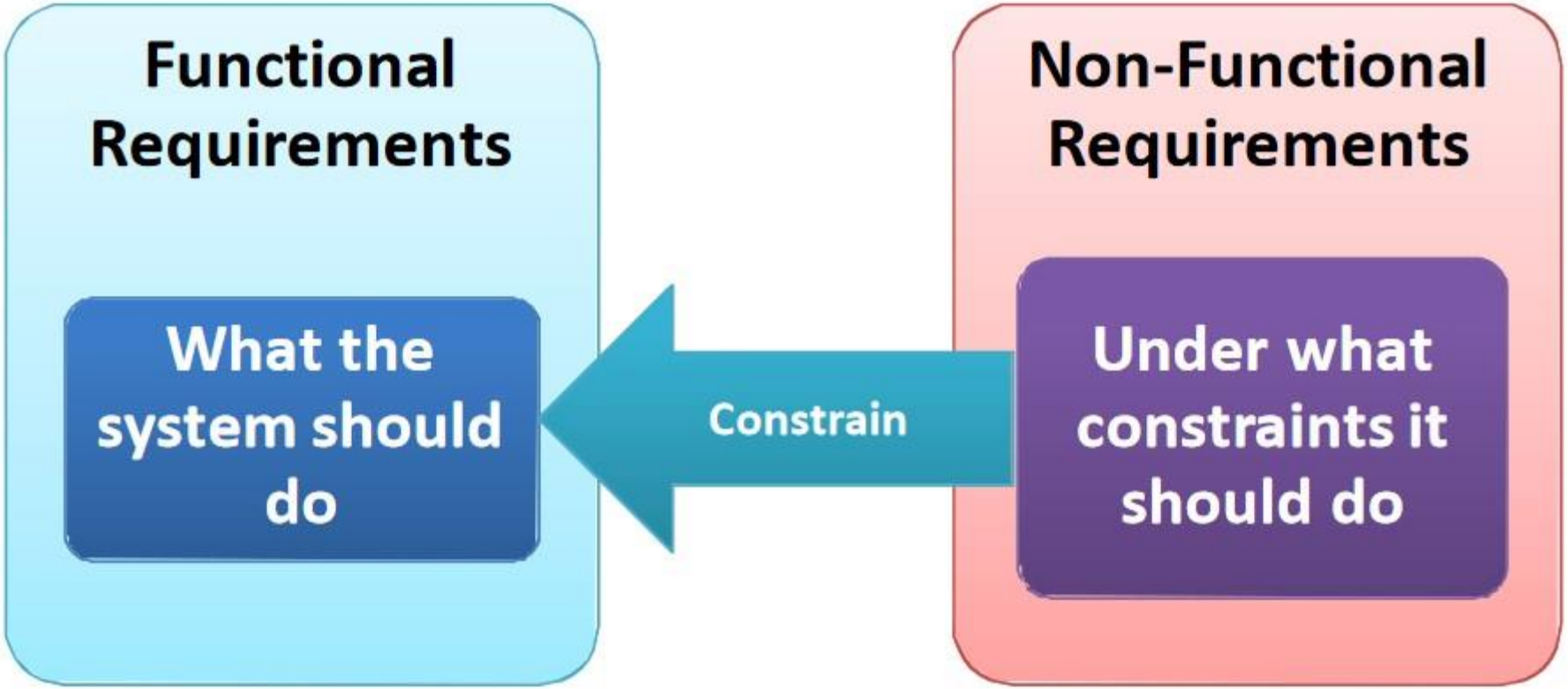
The database will be Oracle 8i

The system will run 7 days a week, 24 hours a day

An online help system is required

All presentation logic will be written in Visual Basic

Functional + Non-Functional
=
Complete Requirements



Functional Requirement

Functional Requirements (1)

specify **functions** that a system or component must be **able** to **perform**

describe what the system should **do**

things that can be captured in **use cases**

things that can be **analyzed** by drawing sequence diagrams, statechart, etc.

will probably **trace** to individual chunks of a program

Functional Requirements (2)

A functional requirement is **testable**

- fold this activity right into the writing

A general rule is: a functional requirement is a **“shall statement”**

Functional Requirements (3)

These can be high level or low level (generally we're at high level in this class)

High level: The system shall charge users credit cards for purchases

Low level: The system shall validate all passwords contain upper and lowercase characters and one number

Functional Requirements (4)

Are **testable**

- your “test lead” should be collecting sets of basic tests generated for each requirement linked to the requirement
  - this is part of writing the requirement!

Should be **one thing** (not multiple). (Because a requirement is a single entity… it passes or fails as one piece)

Should have a **source** (who/what decided this was required)

Should have a **priority**

Functional Requirements (5)

Should not be **a design choice**

- The system shall store user information including name, DOB, address and SSN. <-- **Good**!
- The system shall store user information in an Oracle database including name, DOB, address, SSN. <-- **bad**

Must have **a unique ID**.

- When testing you need to reference REQ-1 or REQ-287. Multiple things cannot be labeled REQ-1.

Functional Requirement Prioritization

Must have requirement	1
Should have requirement Could have requirement	2
Wish list requirement	3

FR- Prioritization Logic

M - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success.

S - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary.

C - COULD: Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit.

W - WISHLIST: Represents a requirement that stakeholders have agreed will not be implemented in a given release, but may be considered for the future. (note: occasionally the word "Would" is substituted for "Won't" to give a clearer understanding of this choice)

Examples of Functional and Non-Functional Requirements

Video over IP
Conference Calling

Functional Requirements

- Add Participant
- Count Participants
- Drop Participant
- Lock Call to New Participants
- Summon Operator
- Mute microphone

Non-Functional Requirements

- Voice and Video Quality
- Reliability
- Availability
- Ease of Use
- Cost
- Localization

Functional Requirements

A **Functional Requirement**:

- is a statement of what a system must do (#1)
- is measured in "yes" or "no" terms
- usually employs the word "shall"

Examples:

Add Participant

"The software shall display an option to add a participant"

Summon Operator

"The software shall summon the operator if the participant clicks the Operator Help icon."

Non Functional Requirement

Non-functional Requirements (1)

- Define the overall **qualities** or **attributes** of the resulting system
- Place **restrictions** on the product being developed, the development process, and specify external constraints that the product must meet.

Examples of NFR include safety, security, usability, reliability and performance requirements.

May be more **critical** than functional requirements

- If these are not met, the system is useless!

Types of Non Functional Requirements

- IEEE-Std 830 – 1993
- (Jacobson, 1999)
- Ian Sommervile
- Nielsen Model
- Mc Call
- Doll & Torkzadeh

Non-Functional Requirement Types also called “ilities”

IEEE-Std 830 – 1993

Performance requirements:

‹1. Interface requirements

‹2. Operational requirements

‹3. Resource requirements

‹4. Verification requirements

‹5. Acceptance requirements

Documentation requirements:

1. Security requirements

‹2. Portability requirements

‹3. Quality requirements

‹4. Reliability requirements

‹5. Maintainability requirements

‹6. Safety requirements

(Jacobson, 1999)

Usability

Reliability

Performance

Supportability

Implementation requirements

Interface requirements

Operations requirements

Packaging requirements

Legal requirements

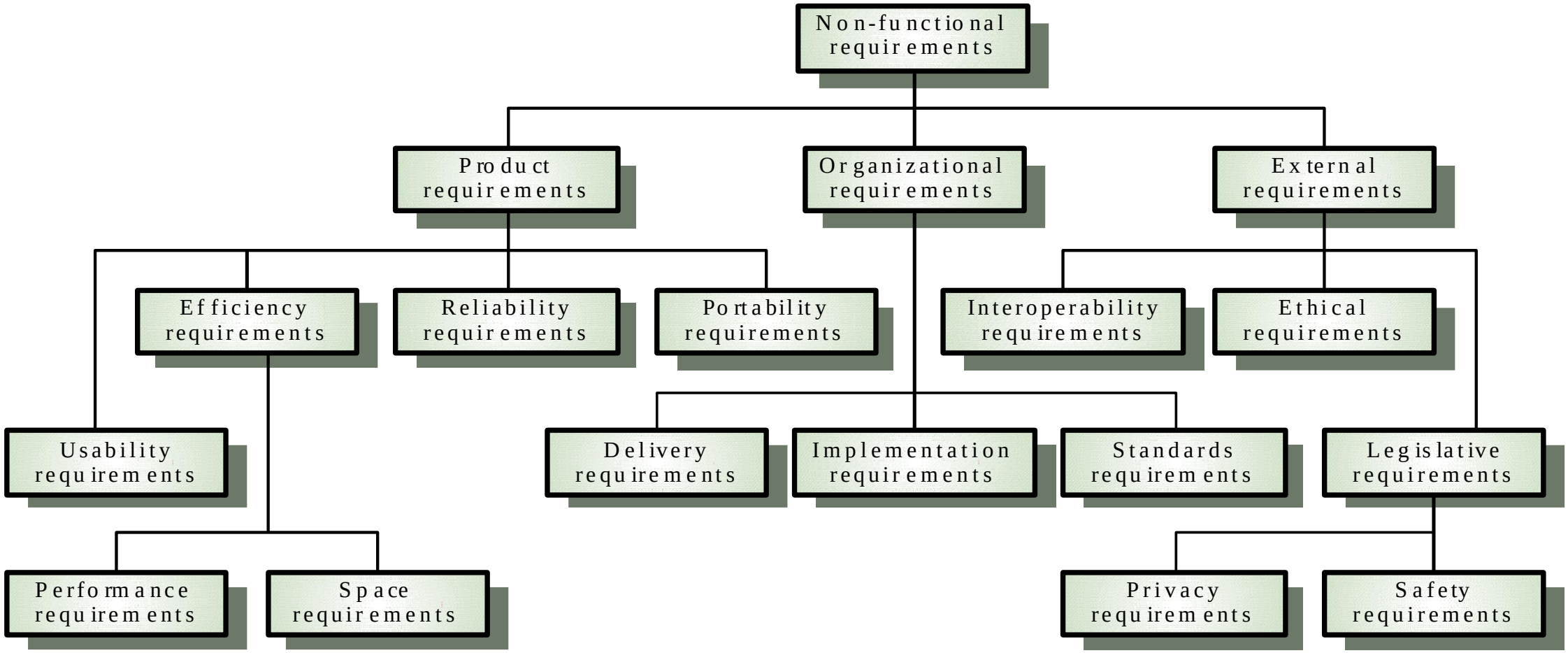
Ian Sommervile

Product requirements:	specify that the delivered product *must behave* in a particular way *e.g. execution speed, reliability, etc.*
Organisational requirements:	are a consequence of *organizational policies* and procedures *e.g. process standards used, implementation requirements, etc.*
External requirements:	arise from *factors which are external* to the system and its development process *e.g. interoperability requirements, legislative requirements, etc.*

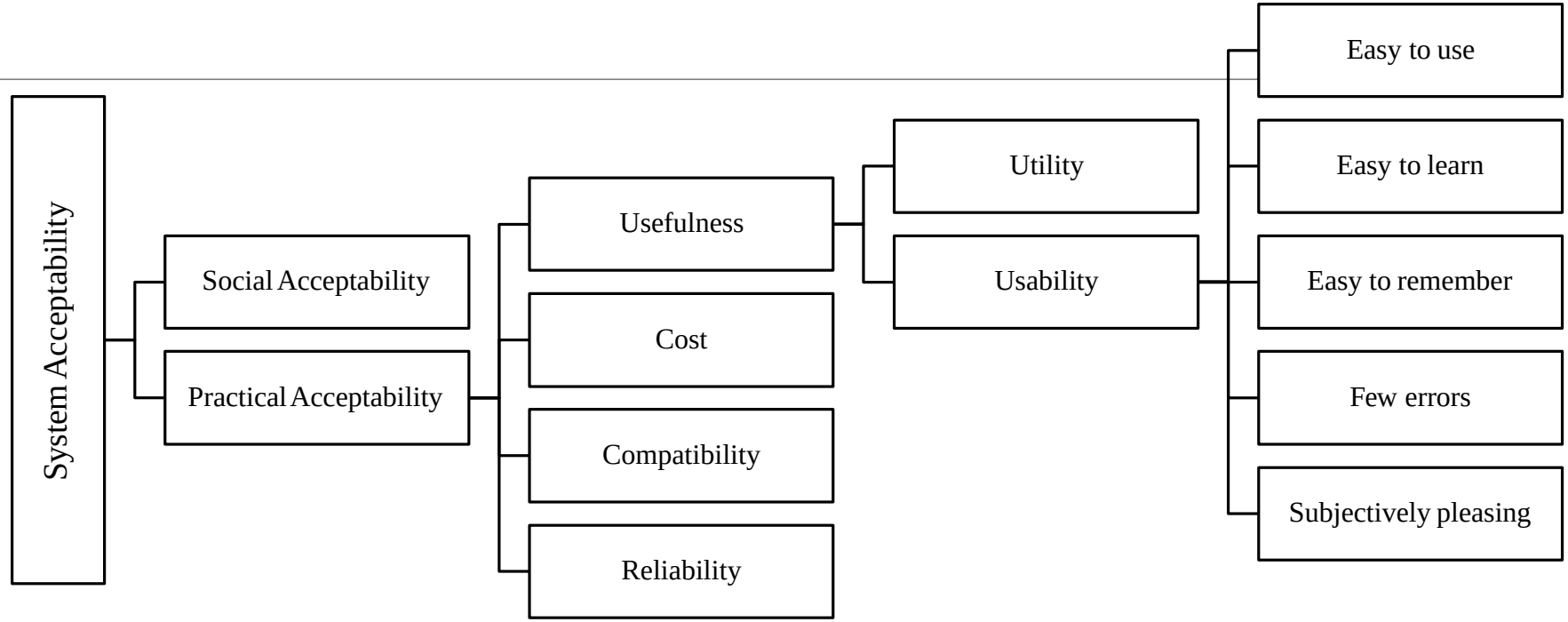
Examples of Non-functional Requirements by Ian Sommervile

Product requirement	The System service X shall have an availability of 999/1000 or 99%. This is a reliability requirement which means that out of every 1000 requests for this service, 999 must be satisfied.
Organisational requirement	The *system development process* and deliverable documents shall conform to the process and deliverables defined in *XYZCo-SP-STAN-95.*
External requirement	The system shall provide facilities that allow any user to check if personal data is maintained on the system. *A procedure must be defined and supported in the software that will allow users to inspect personal data* and to correct any errors in that data.

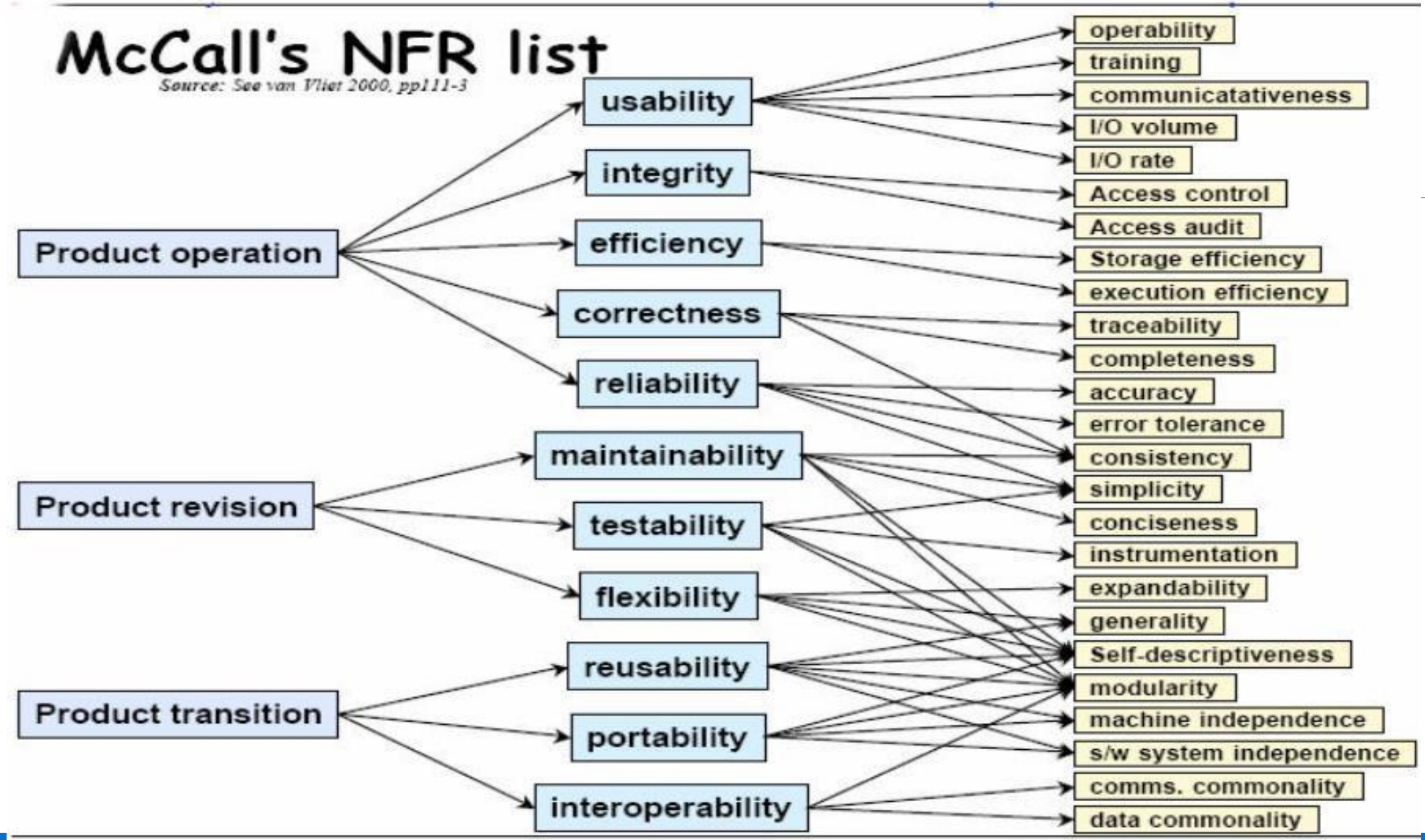
Types of Non Functional Requirements Ian Sommervile



Nielsen Model



System Acceptability Model - Nielsen Model


McCall's NFR list
Source: See van Vliet 2000, pp111-3
Product operation
Product revision
Product transition
usability
integrity
efficiency
correctness
reliability
maintainability
testability
flexibility
reusability
portability
interoperability
operability
training
communicatativeness
I/O volume
I/O rate
Access control
Access audit
Storage efficiency
execution efficiency
traceability
completeness
accuracy
error tolerance
consistency
simplicity
conciseness
instrumentation
expandability
generality
Self-descriptiveness
modularity
machine independence
s/w system independence
comms. commonality
data commonality

Doll and Torkzadeh

Content

C1. Does the system provide the precise information you need?
C2. Does the information content meet your needs?
C3. Does the system provide reports that seem to be just about exactly what you need?
C4. Does the system provide sufficient information?

Accuracy

A1. Is the system accurate?
A2. Are you satisfied with the accuracy of the system?

Format

F1. Do you think the output is presented in a useful format?
F2. Is the information clear?

Ease of Use

E1. Is the system user friendly?
E2. Is the system easy to use?

Timeliness

T1. Do you get the information you need in time?
T2. Does the system provide up-to-date information?

Table 1. Doll and Torkzadeh's End-User Computing Satisfaction Measures

Issues with Natural Language NFRs

While useful in everyday interactions, natural language is fertile ground for a number of issues relating to requirements (functional as well as non-functional) including:

- Weak words
- Unbounded lists
- Implicit Collections
- Ambiguity
- Issues around verb choice, semantics, and grammar

Examples of Natural Language NFRs

Order processing must be fast

Make the web site software reliable

The audio software must reproduce music nearly perfectly

The software must support at least 25 users

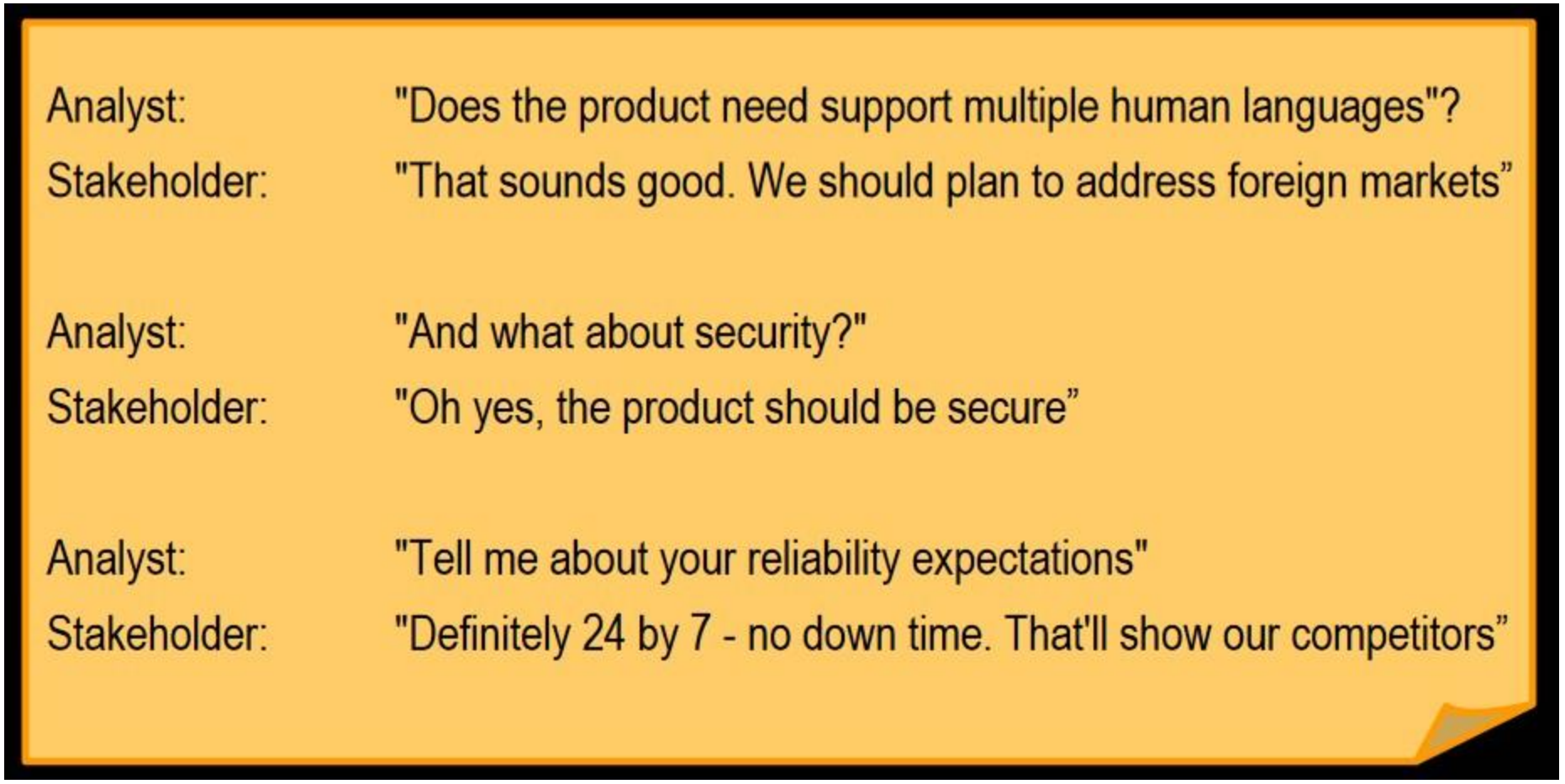
The configuration software should be intuitive to use

Do you see any issues with these requirements?

Issues Identified

1. Order processing must be fast
   - How long is "fast"? Seconds, minutes or hours? Can we test "fast"?
2. The software must support at least 25 users
   - What is the meaning of "support"? Are these concurrent users or not?
   - How many is "at least" 25 users? 26 users? 200,000 users?
3. Make the web site software reliable
   - What is "reliable"? Can we test for it?
4. The configuration software should be intuitive to use
   - "should" implies optionality
   - What does "intuitive" mean? It is subjective (reader dependent)
5. The audio software must reproduce music nearly perfectly
   - What does "nearly perfectly" mean? An audiophile will have a different opinion than a causal listener.

Elicitating the NFRs



Actor/ Stake holder	Use Case (Functional Requirement)	Question no	Question for NFR	Question Answer (Elicited NFR)	Category of NFR
User	Search	NFRQ1	How much time it takes to give Search result	Less than 10 second	Performance
User	Search	NFRQ2	How many ways of searching	Full and partial match word	Flexibility
User	Search	NFRQ3	Autosuggestion is needed when searching	When writing for searching show related work	Usability
User	Login	NFRQ4	How much time it takes for login	Less than 30 sec	Performance
User	Login	NFRQ5	What is the user friendliness needed	Show message if submit without user name or password	Usability
User	Logout	NFRQ6	How much time it takes for logout	Less than 30 second	Performance
Use	Create Account	NFRQ7	How much easy it is to create account	Use drop down box to select relevant option	Usability

CHECKLIST FOR NFR ELICITED NFR

NFR → / FR ↓	Performance	Flexibility	Usability	Modifiability	Privacy	Legal issue	Security
Search	✓	✓	✓				
Login	✓		✓				
Logout	✓						
Create Account			✓				
Update Account				✓	✓		
Handle Payment		✓				✓	
Process Sale	✓						
Delete Account		✓					✓
Handle Coupon						✓	
Add Item	✓	✓					
Delete Item		✓					
Update Item			✓				
Give User Privileged							✓
Read Credit Card	✓		✓		✓		
Print Receipt	✓		✓				
Read Barcode	✓		✓				
Generate Barcode					✓		✓
Calculate Total	✓						
Check Price		✓					
Check Product			✓				